What is Claimed is:

1. A method for transparently maintaining cache coherency when debugging a multiple processor system with common shared memory, the method comprising the steps of:

creating a software memory map of the memory usage of a plurality of processors in the system to be debugged denoting in the software memory map whether or not each processor of the plurality of processors has a cache;

activating a first debug session associated with a first processor of the plurality of processors and at least a second debug session associated with a second processor of the plurality of processors;

detecting a write request to a shared memory location by the first debug session;

passing the write request initiated by the first debug session to the first processor for execution;

searching the software memory map for a first plurality of processors that have read access to the shared memory location;

broadcasting the write request to the first plurality of processors; and

performing cache coherency updates in response to the write request in each of the first plurality of processors.

2. The method of Claim 1 wherein the step of broadcasting the write request comprises indicating that the write request is intended for maintaining cache coherency as opposed to a normal write request.

3. The method of Claim 2 wherein the step of performing comprises using cache coherency capabilities, if any, of a processor in response to the write request intended for maintaining cache coherency.

4. The method of Claim 2 wherein:

the step of creating comprises denoting in the software memory map the shared memory locations that contain program instructions;

the step of passing the write request additionally comprises the step of determining that the shared memory location contains a program instruction; and

the cache is an instruction cache.

5. A method for transparently maintaining cache coherency when debugging a multiple processor system with common shared memory, the method comprising the steps of:

creating a software memory map of the memory usage of a plurality of processors in the system to be debugged denoting in the software memory map whether or not each processor of the plurality of processors has a cache;

activating a first debug session associated with a first processor of the plurality of processors and at least a second debug session associated with a second processor of the plurality of processors;

detecting a write request to a shared memory location by a first debug session;

if the first processor associated with the first debug session has write access to the shared memory location

then

selecting the first processor to perform the write request;

else performing the following steps a-b:

a. searching the software memory map for a second processor with write access to the shared memory location;

b. selecting the second processor to perform the write request;

passing the write request initiated by the first debug session to the selected processor for execution;

searching the software memory map for a second plurality of processors that have read access to the shared memory location;

broadcasting the write request to the second plurality of processors; and

performing cache coherency updates in response to the write request in each of the second plurality of processors.

6. The method of Claim 5 wherein the step of broadcasting the write request comprises indicating that the write request is intended for maintaining cache coherency as opposed to a normal write request.

7. The method of Claim 5 wherein the step of performing comprises using cache coherency capabilities, if any, of a processor in response to the write request intended for maintaining cache coherency.

8. The method of Claim 5 wherein:

the step of creating comprises denoting in the software memory map the shared memory locations that contain program instructions;

the step of passing the write request additionally comprises the step of determining that the shared memory location contains a program instruction; and

the cache is an instruction cache.

9. A software development system, comprising:

a memory storage system holding a software development tool program;

a host computer connected to the memory storage system, the host computer operable to execute the software development tool program;

a test port for connecting to a hardware system, the hardware system being comprised of multiple processors with common shared memory and operable to execute an application program; and

wherein the software development tool is operable to support debugging of the application program executing on the hardware system using a method for transparently maintaining cache coherency when debugging a multiple processor system with common shared memory, the method comprising the steps of:

creating a software memory map of the memory usage of a plurality of processors in the system to be debugged denoting in the software representation whether or not each processor of the plurality of processors has a cache;

activating a first debug session associated with a first processor of the plurality of processors and at least a second debug session associated with a second processor of the plurality of processors;

detecting a write request to a shared memory location by a first debug session;

if the first processor associated with the first debug session has write access to the shared memory location

then

selecting the first processor to perform the write request;

else performing the following steps a-b:

a. searching the software memory map for a second processor with write access to the shared memory location;

b. selecting the second processor to perform the write request;

passing the write request initiated by the first debug session to the selected processor for execution;

searching the software memory map for a second plurality of processors that have read access to the shared memory location;

broadcasting the write request to the second plurality of processors; and

performing cache coherency updates in response to the write request in each of the second plurality of processors.

10. A digital system, comprising:

multiple processors with common shared memory for executing an application program; and

wherein the application program was developed with a software development system using a method for transparently maintaining cache coherency when debugging a multiple processor system with common shared memory, the method comprising the steps of:

creating a software memory map of the memory usage of a plurality of processors in the system to be debugged denoting in the software representation whether or not each processor of the plurality of processors has a cache;

activating a first debug session associated with a first processor of the plurality of processors and at least a second debug session associated with a second processor of the plurality of processors;

detecting a write request to a shared memory location by a first debug session;

if the first processor associated with the first debug session has write access to the shared memory location

then

selecting the first processor to perform the write request;

else performing the following steps a-b:

a. searching the software memory map for a second processor with write access to the shared memory location;

b. selecting the second processor to perform the write request;

passing the write request initiated by the first debug session to the selected processor for execution;

searching the software memory map for a second plurality of processors that have read access to the shared memory location;

broadcasting the write request to the second plurality of processors; and

performing cache coherency updates in response to the write request in each of the second plurality of processors.